POCKET COMPUTER NEWSLETTER

(c) Copyright 1984 POCKET COMPUTER NEWSLETTER

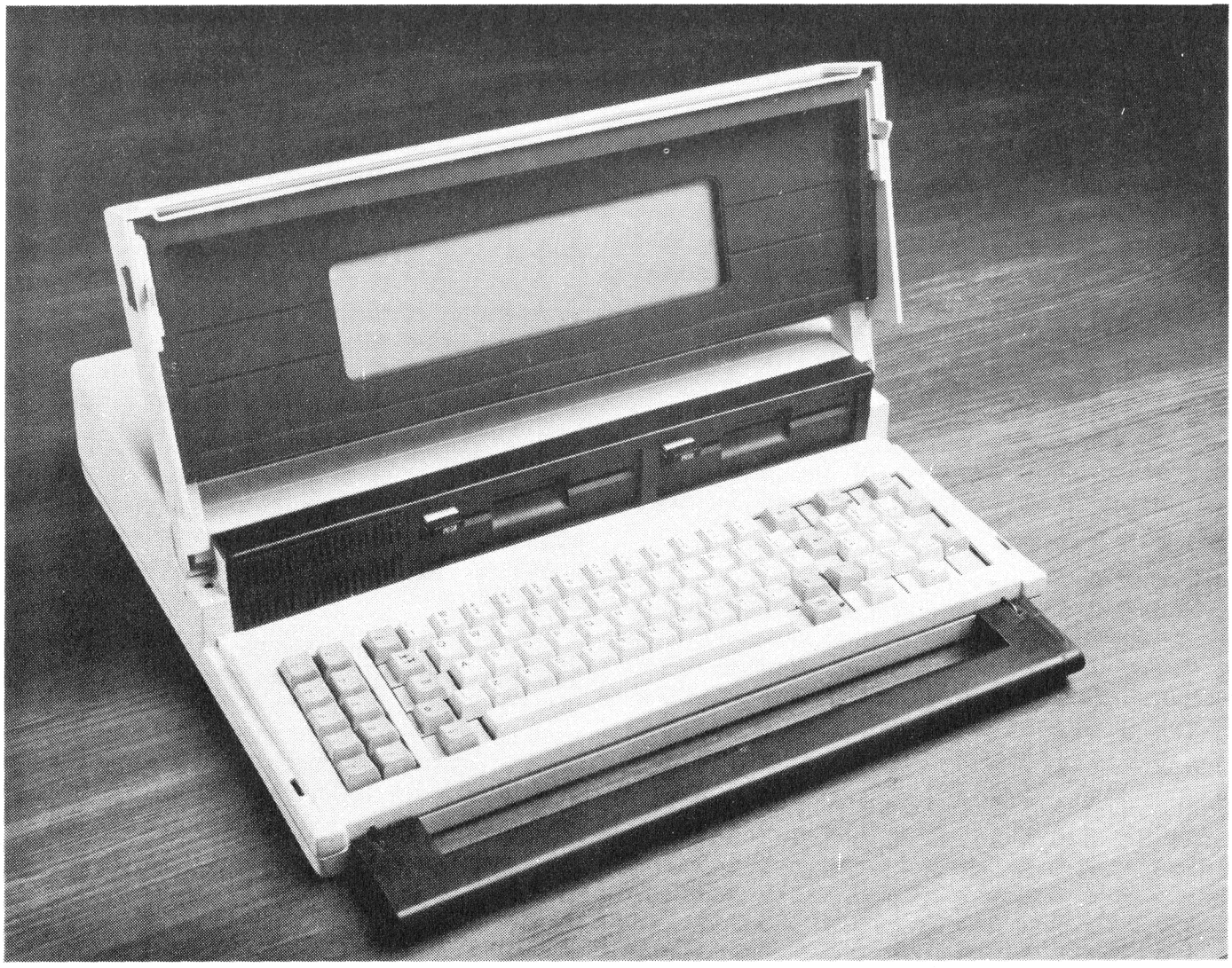
Issue 31 - January/February

NEW PORTABLE DUBBED "THE COMMUTER"

Visual Computer Incorporated of Marlboro, Massachusetts, has announced the design of a new IBM-compatible personal computer that may provide direct competition to the recently released Sharp PC-5000.

The new briefcase-sized portable sports a 16-bit 8088 microprocessor, 128K of memory with parity-checking (expandable to 512K), a 5-1/4 inch floppy disk drive (expandable to two drives), an 83 key low profile keyboard which includes function keys, a numeric pad and layout identical to IBM.

The machine measures 18 by 15 by 3 inches and weighs 16 pounds. It is designed to operate from 110- or 220-volt lines.



Another *Personal Information*(TM) product.

The basic unit is supplied without a display. It contains support logic for an 80 x 25 or 40 x 25 display and an RF connector for a monochrome or color TV monitor. It is claimed to be compatible with IBM bit map color graphics.

As an option, The Commuter may be supplied with an 80 column by 16 line LCD display at a cost of $495.

The unit includes a parallel port for connecting to a letter quality or dot-matrix printer and a serial asynchronous RS-232C port. A connector is also provided for expanding the computer using an IBM expansion chassis.

The Commuter will use MS-DOS (2.1) as an operating system. Software for the IBM PC is said to be compatible with the machine.

For additional information contact *Visual Computer Inc., 135 Maple Street, Marlboro, MA 01752.*

THE SHARP PC-5000 PORTABLE COMPUTER

The new Sharp PC-5000 Portable Computer has finally arrived. Sharp has certainly packed a lot into this relatively small package. The machine really has the power of a conventional desktop-sized computer. However, it is all compressed into a portable unit the measures close to a 13 by 12 by 3-1/2 inches. When you read about all that can be fit within these short dimensions, I think you too will be impressed.

The basic unit without options weighs just 12 pounds. However, there is room within the case of the computer to install a printer and a modem. The addition of these options can add another 4 to 5 pounds.

The front cover of the PC-5000 lifts to reveal a full typewriter-style keyboard containing 72 keys. Eight of these keys are user definable "softkeys."

The underside of this cover contains an 80-column by 8-line liquid-crystal display (LCD). The 640 by 80 pixels on this LCD can be used to display text or graphics. There are 128 text characters/symbols and and equal number of graphic elements available for use. The angle of the cover can be adjusted to one of several possible positions to facilitate comfortable viewing. In addition, an LCD contrast control allows the user to customize the contrast of the display to compensate for actual lighting conditions.

Memory Everywhere

The PC-5000 is loaded with memory. However, because of its sophisticated modes of operation, actual user RAM appears to be used up rapidly!

There are 192 kilobytes of internal ROM installed. This apparently stores the MS-DOS operating system, various system utility routines and part of the highly sophisticated Microsoft BASIC interpreter. Somewhat surprisingly, there is another separate 64 kilobyte ROM containing much the the Microsoft BASIC interpreter. It resides in an expansion slot and usurpes 64 kilobytes of RAM address space. Furthermore, in order to use the BASIC package, you need to have a bubble memory unit plugged in for use as a mass storage device. Now this seems like an awful lot of memory just to run BASIC. However, you may have never seen the likes of this kind of BASIC before. It is undoubtably one of the most powerful dialects of the language available on anything short of a large scale mainframe computer! More about it later.

Perhaps the most intriguing aspect of the PC-5000 is its use of 128 kilobyte bubble memory cartridges. These devices measure about 3 by 2-1/2 by 3/8 inches. They plug directly into a slot on the top of the unit. (The slot is equipped with a flip-up/down cover so the cartridge is invisible when not being inserted or removed.) These bubble memory cartridges are treated exactly as though they were a combination of a floppy disk and its disk drive! In fact, when using the MS-DOS operating system, the bubble memory is treated as drive A (or, by swapping bubble memory cartridges, as drive B). The bubble memory modules are small and slim. They fit easily, for instance, into a shirt pocket. Thus, you can carry several with you in lieu of floppy disks and their associated drives. (But, if you want to use floppies, you can purchase a 5-1/4 inch dual-drive unit as an external option.)

Now the list price of these 128 kilobyte bubble memories is currently rather steep: $269.00 U.S. list. However, you receive one with the PC-5000. It is supplied as part of a software package that includes a text editor program called SuperWriter, published by Sorcim. (Sorcim is a firm that has been producing commercial software for desktop computer. The firm is now planning to provide software support for a number of portable computer systems.)

The module containing Super- Writer has about 24K left over on which you can store your own programs and/or data. However, since 24K can be used up rapidly on a system such as this, you may want to consider obtaining one or two more bubble cartridges to use for storage of your programs and data. (While this may seem expensive when compared to purchasing a couple of floppy disks, remember their use saves the initial costs of the disk drives!)

While bubble memory cartridges cost quite a

bit at the moment, their increasing use should drop the price dramatically in coming years. It would not be surprising to see 128 kilobyte bubble cartridges selling in the range of $50 - $75 within a few years.

When you add it all up, you find that the little PC-5000 comes equipped with an astounding 512 kilobtyes of combined memory, distributed as follows: 192K system ROM, 64K BASIC ROM, 128K user RAM and 128K bubble memory. That is a lot of memory to pack into a portable unit.

Where Does It All Go?

At first, this amount of memory sounds like a staggering sum. However, it turns out that much of this memory is already dedicated to system uses. Thus, the 192K of system ROM and 64K ROM module devoted to BASIC consumes half of the 512K total. Now it turns out that the version of BASIC utilized can only address a maximum of 64K of memory. Close to half of this address range is apparently devoted to the BASIC operating system. The net result of all this is that you are limited to a maximum of approximately 35 kilobytes of user program/data storage when running BASIC. This is nothing to sneeze at (especially if you are accustomed to squeezing programs into a pocket computer!), but it seems a far cry from making use of that nice 128 kilobytes of RAM that is sitting right there in the machine!

On the other hand, the extremely powerful version of BASIC provided, along with the operating system, allows virtually unlimited program "chaining" and "overlay" capability. Since this is accomplished with the bubble memory unit, complex program overlays take but a second or two. Thus, you can construct huge programs as long as program and/or data blocks are held under about 35K at a time!

There are similar types of limitations when using other software packages. Out of the 128K of system RAM provided in the PC-5000, you can count on a significant portion of this being used by the operating system and/or applications package. For instance, when the word processor Super-Writer is loaded into memory (from the bubble cartridge), you are informed that the text buffer has space for about 24,000 characters. That is the equivalent of about 12 pages of ordinary text. Not an awful lot if you are writing a book. In fact, it seems like a rather chintzy portion of that 128K of RAM. It turns out, however, that the amount of memory available for a text file is limited by the amount of file storage (bubble memory or disk space). About 24K is what is left on the bubble memory that is used to supply the SuperWriter program. If you plug in another bubble memory or use the disk version, you can have much larger text files.

In summary, don't make the assumption that the amount of memory supplied with the unit is a lot more than you will ever need. In fact, its rapid consumption by application programs can be downright disconcerting to beginning users. Chances are you will eventually want to add at least a few more bubble memory units for storage of files or expand to the floppy disk system.

What A BASIC

A customized version of Microsoft BASIC is provided with the PC-5000. It is one of the most powerful, flexible implementations of the BASIC language that you are likely to see short of a large mainframe installation. Just about the only shortcoming is the poor instruction manual that covers this part of the machine's operation.

To give you an idea of the extent of this BASIC's power, here is an alphabeticized list of the reserved words associated with it: ABS, AND, ASC, ATN, AUTO, BEEP, BLOAD, BSAVE, CALL, CDBL, CHAIN, CHR$, CINT, CIRCLE, CLEAR, CLOSE, CLS, COLOR, COM, COMMON, CONT, COS, CSNG, CSRLIN, CVD, CVI, CVS, DATA, DATE$, DEF, DEFDBL, DEFINT, DEFSNG, DEFSTR, DELETE, DIM, DRAW, EDIT, ELSE, END, EOF, EQV, ERASE, ERL, ERR, ERROR, EXP, FIELD, FILES, FIX, FN, FOR, FRE, GET, GOSUB, GOTO, HEX$, IF, IMP, INKEY$, INP, INPUT, INPUT#, INPUT$, INSTR, INT, KEY, KILL, LEFT$, LEN, LET, LINE, LIST, LLIST, LOAD, LOC, LOCATE, LOF, LOG, LPOS, LPRINT, LSET, MERGE, MID$, MKD$, MKI$, MKS$, MOD, MOTOR, NAME, NEW, NEXT, NOT, OCT$, OFF, ON, OPEN, OPTION, OR, OUT, PAINT, PEEK, PLAY, POINT, POKE, POS, PRESET, PRINT, PRINT#, PSET, PUT, RANDOMIZE, READ, REM, RENUM, RESET, RESTORE, RESUME, RETURN, RIGHT$, RND, RSET, RUN, SAVE, SCREEN, SGN, SIN, SOUND, SPACE$, SPC, SQR, STEP, STOP, STR$, STRING$, SWAP, SYSTEM, TAB, TAN, THEN, TIME$, TO, TROFF, TRON, USING, USR, VAL, VARPTR, VARPTR$, WAIT, WEND, WHILE, WIDTH, WRITE, WRITE#, and XOR!

That, my friends, is a powerful implementation that offers something for almost everyone. Yes, you can access serial and random files, do almost anything you want in terms of controlling the LCD screen, call machine language routines, write structured programs, etc. Many of the commonly used keywords are available at the stroke of a single key by using the ALTernate button. (This feature can save a lot of time if you are not a skilled typist.) Furthermore, installed in the PC-5000, this version of BASIC is *fast!*

After perhaps 30 - 40 hours of programming

with this version of BASIC, I have the following complaints: 1. The instruction manual provided with the PC-5000 for this BASIC was obviously done on a "crash" basis. While there is about 150 pages of information describing the instruction set, it looks as though it was literally thrown together with not so much as a single proof-reading. There are countless typographical errors, sections are rather incomplete and difficult to understand, and sometimes missing altogether. For instance, the PRINT# statement, which is crucial to writing to external files, is referenced in parts of the manual (so that you know it is available), but is never presented. Hopefully, Sharp will institute a "crash" program to get this portion of their documentation upgraded to the level of the quality of the machine itself. (It should be noted that some of the other documentation supplied with the unit, especially the SuperWriter material, is excellent.) 2. The RENUMber command seems to lack a facility normally provided - that of being able to renumber just a specific section of a program. This seems to be an oversight also, as you are permitted to start renumbering anywhere in a program. You just cannot specify an ending line number. In other words, once it starts renumbering, it does so until it reaches the end of the program.

In summary, however, this is a powerful, extended version of BASIC that is sure to please most users, especially if future manuals are upgraded so that all of its many features are clearly explained.

SuperWriter

The PC-5000 is supplied with an applications overlay menu that is designed to support several programs designed by Sorcim. You receive one of these programs (supplied on a bubble memory cartridge) with the unit. The program is a text editor or word processor named *SuperWriter*. It is good. And, the manual for it is also good.

I am not going to go into a lot of detail on the operation of SuperWriter, other than to say that it is a powerful, flexible, menu-driven, text editor. It permits formatting of the output (including full justification, pagination, footers, etc.), movement of text, merging operations, and so forth. The program is supplied with a comprehensive manual plus a short "ten-minute tutorial" booklet and a handy pocket reference card. If you write letters, reports, articles, even books, you will find a lot of use for this program. It, as all other application programs do, will undoubtably have idiosyncracies that may annoy you slightly. But, all-in-all, it will get the job done for most people.

The Applications System

The SuperWriter program is just one in an optional integrated set of programs that may be obtained for the PC-5000. An understanding of how these programs can interrelate is important to gaining a full appreciation of the true power provided in this portable machine.

Several other application packages may be purchased as options. Currently, these include: SuperCalc, SuperPlanner and SuperComm. As you might surmise, SuperCalc provides spreadsheet capabilities. SuperPlanner provides appointment and database functions. SuperComm is for managing communications when using a modem.

Now having all of these capabilities available in a portable unit might be enough in itself. But wait until you hear the rest.

Access to each of these programs is provided through a menu system. For instance, if you want to use SuperCalc, you make sure a bubble memory containing the program is installed and select the desired menu option. (If the correct memory module is not installed, you will be so-informed by the menu program.) Once the SuperCalc program is on-line, new menus enable you to select the operations you currently desire. One of the options provided is the ability to copy data from a spreadsheet (or other Super application programs) into what Sharp calls "scratch-pad" memory.

You guessed it! When you switch from one Super applications program to another, the scratch-pad memory is not altered. Thus, you can transfer information from a spreadsheet into a SuperWriter text file! Or, from a SuperComm file into SuperWriter. And, vice-versa. Do you get the picture? The system allows integration of data between application programs! Now that is Super!

MS-DOS

Files under Super applications programs, BASIC, and presumably other language options in the future, all run under the built-in Microsoft-DOS system. This is a powerful file-handling and directory system that is accessible to the user. You can create files (on external memory devices such as bubble memories of floppy disks), examine their contents, duplicate, merge, delete, rename, etc. You can also create hierarchal directories and these can be nested.

Once again, the system is powerful. My only complaint in this area is the poor documentation. They provide a manual, but it looks as though it was thrown together in the greatest haste. If Sharp doesn't rewrite this portion soon, someone else may make a fortune producing supplemental manuals for the PC-5000.

Other Features

The tape cassette interface is relatively fast. You

can load an 8K BASIC program in about one minute. (It would take about 10 minutes to load the same amount into a PC-1500.) This means that the tape interface is a practical way to archive programs and data. You can keep your working copies on a bubble, knowing that should something ever happen, you won't have to take half-a-day to restore the information from audio tape cassettes. It may also be a good way for people to distribute programs. By the way, at least when using BASIC, the PC-5000 provides an option of writing programs to external devices, such as via the tape cassette interface, that cannot be listed or edited, only executed.

The optional printer ($395.00 extra) is amazing. In the first place, it is truly quite. It is the first printer I have ever seen on a portable computer that I would not be embarrased to operate in, say, a public library. However, it has its limitations and is not for everyone. It is relatively slow. It is designed to be sheet-fed. A long report means many stops to insert fresh sheets of paper. It requires special thermal paper or the use of special (expensive) ribbons when used with regular paper. If you need to obtain hardcopies of information when you are on the road, the printer is a must. However, if you can wait until you get back to your base of operations in order to get hardcopies, you might be better-advised to select another alternative: purchase a separate external printer and drive it via the PC-5000's built-in RS-232 serial interface!

The PC-5000 does not use nickel-cadmium batteries. It uses a special, sealed, lead-acid package that is surprisingly small. This will power the unit for about 8 hours of operation if the optional printer is not used. Using the printer considerably reduces the battery life. A warning light comes on when the battery needs recharging. A recharger, supplied with the PC-5000, may be used to continuously power the unit, thereby keeping the battery continuously recharged until needed for field-portability.

Looks Like A Winner

All-in-all, the PC-5000 looks like a lot of value, at this time, for its $1995.00 list price. While it will surely face stiff competition in the future, the fact that it is one of the first out of the gate will surely be an important factor in the race to gain popular acceptance. It contains a lot of memory, a powerful version of BASIC, the popular Microsoft operating system, and a menu-driven means of integrating a line of optional application packages. Right now it appears to be the lowest-priced unit with such capabilities. Put all that together and it sure looks like a potential winner!

LITTLE EDITOR

H. David Jackson, 126 Smithfield Drive, Endicott, NY 13760, provides this version of a text editor for PC users.

Introduction

LED is the abbreviation I have assigned to a program called the Little EDitor. This program may be used to input data sets and later edit, alter or delete that data as desired. Although it is small, it still contains many of the functions found in editors that run on large mainframe computers. It is somewhat slow because it is written in BASIC. But, what it lacks in speed it makes up for in function. The use of BASIC also allows the user to easily add to, delete or modify its operation.

LED is considered user friendly as commands are entered under function key control rather than having to type in a directive. Only data and certain parametric information, such as data set names, have to be typed in.

This program was developed on a Radio Shack PC-2 with an 8K memory module. A 4K module can be used, but the maximum size of the data set that could be handled will be considerably reduced. The program requires a printer/cassette interface for saving, restoring and printing the data sets.

Data Set Size

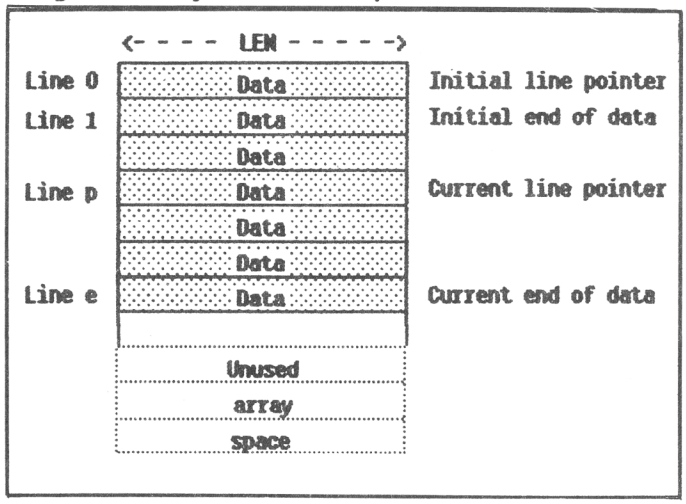
The data set size is set by the user immediately upon entry to the program. A variable called NUM will be requested. This value is the maximum number of lines that will be entered. The largest value that may be used is 255. LEN will then be requested. This is the maximum length of the lines that will be used. The largest allowable value here is 80. These two variables are used to dimension an array that stores the data. Plan your data set size carefully as the array cannot be re-dimensioned without losing data already stored. Also, the entire array is saved on tape when requested, no matter how much or little data has actually been entered. This takes some time if maximum limits have been chosen and wastes time if the space is not needed. Of course, it is not necessary to fill all lines reserved. The array can contain unused space.

The size of the array (NUM times LEN) cannot exceed the amount of storage available. If it does, an error message will be displayed. Use the MEM command prior to running the program to see how much memory is available. With an 8K module there will be approximately 7500 bytes, 3400 bytes with a 4K module, provided that memory is not

allocated for other purposes.

Regardless of the array size, internal pointers to the current line being entered or edited, as well as the last line in the data set, are maintained by the LED program. These are initially set to zero and one respectively. Using the end of data (EOD) allows the line pointer to be more easily set to selected locations without having to traverse large amounts of unused lines. Thus, the initialized data set is two lines long. Additional lines may be added as needed up to the extent of the array size (NUM) that was originally specified.

Diagram *Array and Data Space Allocations.*



Data Display and Modes

The amount of data that is seen at one time is limited to the 26 characters of the display screen. The length of a line, however, can be up to the LEN orginally specified. In order to accomodate this, two types of displays are used. The first is the input display. This is used when the program has been placed in the input mode. It is denoted by the presence of a question mark (?) at the left of the screen. In this case, the normal responses and editing associated with the use of the INPUT statement in BASIC apply. The display scrolls left one character at a time when the right end of the screen is reached.

The second type of display is associated with the edit mode. It is denoted by a flashing cursor. While in this mode, all of the function keys that will be described later are operational. Key usage in this mode is a little different. Various modes are used to tell the program what to do. The current mode(s) are indicated by the cursor. These modes are summarized as follows:

SHIFT MODE - This mode is similar to the regular shift key function used in the INPUT section. It, however, also modifies the commands associated with some of the cursor movement keys. This mode is entered by pressing the SHIFT key. If the current line is null (contains no data), then the program will enter the input mode (a ? appears on the left of the screen). This allows data to be entered faster than when in the edit mode. The program will remain in this mode for each line entered until a non-null line is found or the ENT key is pressed without entering any data. In either case, the program will revert back to the edit mode.

ALTERNATE MODE - This mode also changes the command that is associated with function and cursor movement keys as described later. This mode is set by pressing the Reserve mode change (up and down solid triangle) key.

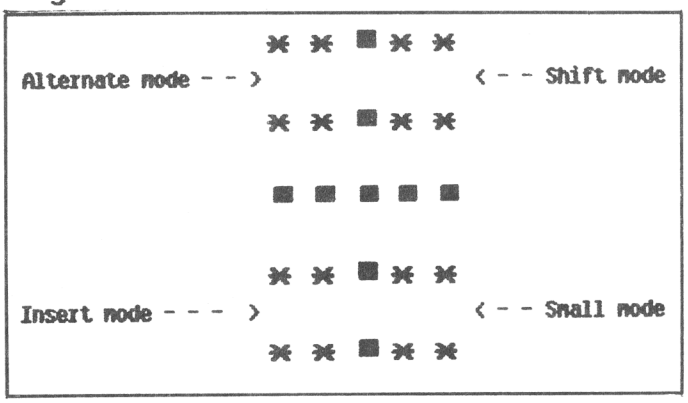
(The above two modes are automatically reset after another key is pressed. That is, they are only in effect for one key stroke.)

INSERT MODE - This mode indicates that characters are to be inserted at the cursor position instead of overlaying those that are already there. This mode is toggled by pressing the MODE key.

SMALL MODE - As the name implies, characters that are overlaid or inserted will be in lower case. This mode is toggled by pressing the SML key.

The cursor is used to indicate the current mode status. This is done by placing small blocks in each corner of the cursor. If just a flashing character (or lone underline when the position is blank) is observed, then none of the above modes are set. The accompanying diagram shows which corner blocks are active for the various modes. Note that more than one mode may be set at a time.

Diagram *Cursor Mode Indicators.*



In addition, when in the edit mode, the cursor and display positioning is slightly different than that used by the input mode. When the cursor leaves the right side of the display in the edit mode, it will be positioned to the middle of the display instead of just scrolling left one. This gives the user a more complete view of the line being edited and allows one to see data on both sides of the

cursor. The same occurs, when moving to the left, unless the cursor is in the first position.

Normal Function Keys

The following functions or commands are executed by pressing the specified key when editing in the normal mode. A beep indicates that the depression of a function key (F1 - F6) has been recognized by the program.

F1 - LOAD: This command will load a data set that was previously stored on tape. It is necessary to position the tape to the proper spot. Then enter the name of the data set that is to be recovered. The program restores the array parameters that were saved along with the data. Pressing the ENT key by itself causes the command to be ignored.

F2 - ADD: This command will add a single null line immediately after the line on which the cursor currently resides. If you listen carefully, an audio click may be discerned, indicating that one line has been added.

F3 - CLEAR MARKS: This command will clear any marks that may have been placed on selected line(s) of data.

F4 - MARK DATA: The first press of this key will *mark* the line at the cursor position for later reference. This line will be highlighted by reverse video. A second press will mark a group of lines, all of which will be denoted by reverse video. The group comprises all lines between and including the marked lines. A marked line or group of lines may be moved, deleted or printed. Marks may also be used to limit changes to a selected line or group.

F5 - MOVE: Pressing this key causes the marked area to be moved after the line on which the cursor is currently located. The marked data is not erased. Thus, this directive may be used to duplicate information at various locations. (To erase data see alternate F2.)

F6 - FIND DATA: This key is used to indicate that a word or phrase is to be searched for in the data beyond the current cursor or in a marked area. If located, that line will become the current line in the display. If the search is unsuccessful, the line pointer remains the same.

Alternate Function Keys

The following functions will be executed upon pressing the specified key while editing in the alternate mode.

AF1 - SAVE: This command will save the data set and associated parameters. You will have to position the tape, make sure you have set the tape unit to the record mode, and enter a file name for the data set. Pressing the ENT key without other input will cause the command to be ignored.

AF2 - DELETE: This command causes the line that the cursor is on or a marked group of lines to be removed from the data set. A low tone is issued for each line that is deleted.

AF3 - DUPLICATE: This key causes the line that the cursor is on to be duplicated immediately after the line.

AF4 - Not used.

AF5 - PRINT: The entire data set or a marked group of lines is printed. Prior to outputting of the data, the program will prompt for the CSIZE desired by the operator.

AF6 - CHANGE DATA: A global change function. The system prompts for *from* data and *to* data. All cases of the former following the cursor to the end of the data set or within a marked group are modified to the latter. A null *from* entry causes the *to* data to be inserted at the start of each line. A null *to* entry will cause the *from* data to be removed.

Shifted Function Keys

If the shift mode is on, the function keys F1 - F6 will react as though in the input mode. That is, the characters !, ", #, $, % and & will be entered. The shift key is also used to enter @, >, <, ?, ;, : and comma as indicated on the keyboard.

The shift key is additionally used to activate the CLR key. This combination causes the data array to be re-initialized.

If the OFF key is pressed while in the shift mode, the program is terminated.

Cursor Control Keys

The cursor control keys perform in the following manner:

The right arrow causes the cursor to move one character position to the right.

Shift right arrow causes the character beneath the cursor to be deleted. The cursor remains at the same position.

The left arrow causes the cursor to move one character position to the left.

Shift left arrow causes the character to the left of the cursor to be eliminated. The cursor also moves left one position.

The up arrow causes the line pointer to move one line closer to the start of the data set (-1).

Alternate up arrow causes the line pointer to be moved five lines closer to the start of the data set (-5).

Shift up arrow causes the line pointer to be set to the start of the data set. (Note that this cannot be performed on a null line as the shift will cause the program to go into the input mode.)

The down arrow moves the line pointer one line closer to the end of the data set (+1).

Alternate down arrow causes the line pointer to

move five lines closer to the end of data (+5).

Shift down arrow sets the line pointer to the end of data. (Again, this operation cannot be used while on a null line.)

Additional Key Capabilities

The following keys also assist during editing operations:

The RCL key adds five lines after the current line.

Alternate RCL adds 10 lines.

Shift RCL adds 20 lines.

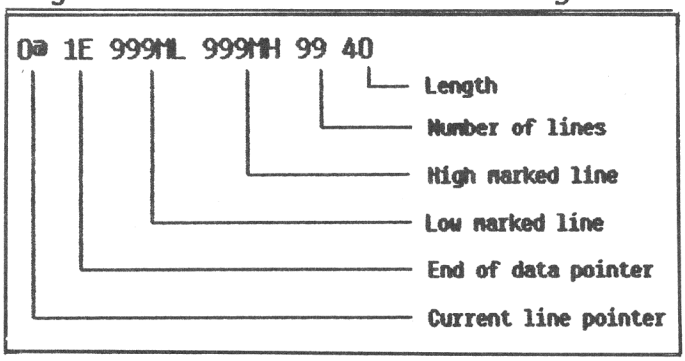
A fast audio click announces the addition of each line.

The DEF key displays the status of all the important pointers. This display uses the format illustrated in the accompanying diagram. (The diagram assumes that the data set initially had an array size whereby NUM=99 and LEN=40. The value 999 in the marked lines positions indicates that no lines are marked.)

Program *Little Editor.*

```
 4:CLEAR :INPUT "
   NUM-";N:INPUT
   "LEN-";O
 5:DIM M$(N)*O:A=
   1
10:L=999:M=999:
   WAIT 0:C=1:P=1
20:POKE 18409,72,
   118,74,0,5,189
   ,255,65,78,78,
   153,8
30:POKE 18421,76,
   119,139,6,72,1
   19,74,0,158,18
   ,154
50:D=0:E=0
55:IF B<=0LET B=0
60:IF B>=ALET B=A
70:Y=LEN M$(B):IF
   Q>25LET P=P+13
   :Q=Q-13
80:IF Q<0AND P>1
   LET P=P-13:Q=Q
   +13
90:IF Q<0LET Q=0
95:IF (P+Q)>YLET
   Q=Y-P+1
100:C=P+Q:PRINT
   MID$ (M$(B),P,
   26)
110:IF B>=LAND B<=
   MCALL 18409
120:GCURSOR 6*Q:IF
   D+E+F+G+((MID$
   (M$(B),C,1)="
   ")OR C>Y)=1
   THEN GPRINT "4
   040404040":
   GOTO 140
130:GPRINT E+G;E+G
   ;0;D+F;D+F
134:IF (Y=0)AND (D
   >0)GOTO 4000
140:K=ASC INKEY$ :
   IF K=0GOTO 100
145:IF K<>32AND K<
   40GOTO 200
150:IF D=0GOTO 170
155:IF K=61LET K=6
   4
160:IF K=32LET K=9
   4
165:IF K>39AND K<4
   8LET K=ASC (
   MID$ ("<>:;,,.
   ?",K-39,1))
170:IF K>64AND K<9
   1AND F>0LET K=
   K+32
175:IF C>YLET M$(B
   )=M$(B)+CHR$ K
   :GOTO 185
180:M$(B)=LEFT$ (M
   $(B),C-1)+CHR$
   K+RIGHT$ (M$(B
   ),Y-C+(G>0))
185:Q=Q+1:GOTO 50
200:IF K=12AND D=0
   LET Q=Q+1:GOTO
   50
210:IF K=8AND D=0
   LET Q=Q-1:GOTO
   50
220:IF K=31LET G=1
   12*(G=0)
225:IF K=2LET F=11
   2*(F=0)
230:IF K=1LET D=7*
   (D=0):GOTO 55
235:IF K=9LET E=7*
   (E=0):GOTO 55
240:IF K=13LET B=B
   +1:P=1:Q=0
245:IF K=10LET B=B
   +1+(4*(E>0)):B
   =B+((A-B)*(D>0
   ))
250:IF K=11LET B=(
   B-1-(4*(E>0)))
   *(D=0)
255:IF K=25LET Y=5
   +(5*(E>0))+(15
   *(D>0)):GOSUB
   3000
260:IF K>16AND K<2
   3AND D>0LET K=
   K+16:GOTO 175
275:IF K=12LET M$(
   B)=LEFT$ (M$(B
   ),C-1)+RIGHT$
   (M$(B),Y-C+(C>
   Y))
280:IF K=8LET M$(B
   )=LEFT$ (M$(B)
   ,C-1-(C<>1))+
   RIGHT$ (M$(B),
   Y-C+1):Q=Q-1
290:IF K=24AND D>0
   GOTO 4
300:IF K=15AND D>0
   END
315:IF K=27THEN
   PAUSE B;"@";A;
   "E";L;"ML";M;"
   MH";N;O
317:IF K<17OR K>22
   GOTO 50
320:BEEP 1
321:ON K-16+6*(E>0
   )GOTO 1000,120
   0,1300,1400,15
   00,1600,2000,2
   100,1800,50,22
   00,2300
1000:INPUT "LOAD-
   ";Z$
1005:IF Z$=""GOTO
   50
1010:CLEAR :INPUT
   #Z$;N,O,A:
   DIM M$(N)*O:
   INPUT #"";M$
```

Diagram *Status Indicators for the LED Program.*



Error Messages

Normal system errors, such as exceeding storage capabilities, are announced by regular system error messages.

Errors associated with the editor itself cause five beeps to be emitted. This occurs if an attempt is made to add more lines than are defined for the array or move an unmarked data item. (Note that moving data may cause the array limits to be exceeded.)

```
      (*):GOTO 10
1200:GOSUB 1850:
      GOTO 50
1300:GOSUB 2500:
      GOTO 50
1400:M=B:IF L=999
      LET L=B
1410:IF M<LLET Z=
      M:M=L:L=Z
1420:GOTO 50
1500:IF L=999THEN
      BEEP 5:GOTO
      50
1505:Y=M-L+1:
      GOSUB 3000:
      IF A+Y>NTHEN
      RETURN
1510:BEEP 2,100,3
      00:IF L>BLET
      L=L+Y:M=M+Y
1520:FOR Z=1TO Y:
      M$(Z+B)=M$(Z
      +L-1):NEXT Z
      :GOTO 50
1600:X=B:Y=A:
      GOSUB 1900:T
      $="FIND-":
      GOSUB 1950:V
      =0:GOSUB 196
      0:IF TLET B=
      Z:GOTO 50
1610:PAUSE "NOT F
      OUND":GOTO 5
      0
1800:GOSUB 1850:M
      $(B+1)=M$(B)
      :GOTO 50
1850:Y=1:GOSUB 30
      00:RETURN
1900:IF L<999LET
      X=L:Y=M:L=99
      9:M=L
1910:RETURN
1950:W$="":PRINT
      T$;:INPUT ""
      ;W$
1955:U=LEN W$:
      RETURN
1960:T=0:FOR Z=X
      TO Y
1970:V=V+1:IF
      MID$ (M$(Z),
      V,U)=W$LET T
      =1:RETURN
1980:IF V<=(LEN M
      $(Z)-U+1)
      GOTO 1970
1990:V=0:NEXT Z:
      RETURN
2000:INPUT "SAVE-
      ";Z$
2005:IF Z$=""GOTO
      50
2010:PRINT #Z$;N,
      0,A:PRINT #"
      ";M$(*):GOTO
      50
2100:X=B:Y=B:
      GOSUB 1900:
      FOR Z=YTO A:
      M$(Z-Y+X)=M$
      (Z+1):NEXT Z
      :A=A-Y+X-1
2105:FOR Z=1TO Y-
      X+1:BEEP 1,2
      55,100:NEXT
      Z:GOTO 50
2110:FOR Z=YTO A-
      1:M$(Z-Y+X)=
      M$(Z+1):NEXT
      Z:GOTO 50
2200:X=0:Y=A:
      GOSUB 1900
2210:INPUT "SIZE-
      ";T:CSIZE T:
      FOR Z=XTO Y:
      LPRINT M$(Z)
      :NEXT Z:GOTO
      50
2300:X=B:Y=A:
      GOSUB 1900:T
      $="FROM-";
      GOSUB 1950
2310:T$="":PRINT
      "TO-";:INPUT
      "";T$
2320:IF U=0THEN
      FOR Z=XTO Y:
      M$(Z)=T$+M$(
      Z):NEXT Z:
      GOTO 50
2330:V=0:GOSUB 19
      60
2340:IF TLET X=Z:
      M$(Z)=LEFT$
      (M$(Z),V-1)+
      T$+RIGHT$ (M
      $(Z),LEN M$(
      Z)-V-U+1):V=
      V+LEN T$+1:
      GOTO 2330
2350:GOTO 50
2500:L=999:M=L:
      RETURN
3000:IF A+Y>NBEEP
      5:RETURN
3010:FOR Z=ATO B+
      1STEP -1:M$(
      Z+Y)=M$(Z):
      NEXT Z:FOR Z
      =B+1TO B+Y:M
      $(Z)="":BEEP
      1,1,1
3020:NEXT Z:A=A+Y
      :RETURN
4000:INPUT M$(B)
4010:D=D*(M$(B)<>
      ""):B=B+(D>0
      ):GOTO 55
```

Notes for Programmers

The following information is provided for those who may wish to add to, modify or remove functions.

Lines	Description
4 - 10	Initialize variables, array.
20 - 30	Set up inverse display routine.
50	Reset shift/alternate switches.
55 - 60	Adjust current line number if outside array bounds.
70 - 90	Determine length of current line, adjust display partition and cursor position if cursor is off the screen.
95	Adjust cursor position to end of line if higher than line length.
100	Determine cursor position, display current partition of current line.
110	If current line is within marked limited, display partition in reverse video.
120 - 130	Display editor cursor at screen cursor location.
134	If null line and shift mode go to input routine.
140	Get key data. Repeat display if no key pressed.
145	Determine if a function key was pressed.
150 - 165	If in shift mode, adjust key data to shifted value.
170	Adjust key data if small mode and alpha key.
175	If cursor is at end of line, add character to end of line.
180	Otherwise, overlay or insert character at cursor position depending on status of mode switch.
185	Increment screen cursor position and go display again.
200	Adjust cursor if unshifted right arrow.
210	Adjust cursor if unshifted left arrow.
220	Adjust insert switch if MODE pressed.
225	Adjust small switch if SML pressed.
230	Adjust shift switch if SHIFT pressed.
235	Adjust alternate switch if up/down triangle pressed.
240	Adjust line number, partition number and screen cursor if ENT is pressed.
245	Adjust line pointer if down arrow pressed.
250	Adjust line pointer if up arrow pressed.
255	Add multiple lines to data set if RCL is pressed
260	If shifted function key, adjust key data and go insert or overlay data in the current line.
275	Delete the character at the cursor if shifted right arrow key.
280	Delete character to left of cursor if shifted left arrow key.
290	Re-initialize if shift and CLR.
300	End if shift and OFF.
317 - 321	Go to appropriate function key routine.
1000 - 1010	Load data routine.
1200	Add a line routine.
1300	Clear marked data routine.
1400 - 1420	Mark a line routine.
1500 - 1520	Move data routine.
1600 - 1610	Find data routine.
1800	Duplicate a line routine.
1850	Add a single line subroutine.
1900 - 1910	Set limits subroutine.
1950 - 1955	Input parameters subroutine.
1960 - 1990	Find data subroutine.
2000 - 2010	Save data routine.
2100 - 2110	Delete a line or group routine.
2200 - 2210	Print data routine.
2300 - 2350	Change data routine.
2500	Reset marks subroutine.
3000 - 3020	Add lines subroutines.
4000 - 4010	Input data subroutine.

Variable	Description
A	End of data pointer.
B	Current line pointer.
C	Line cursor position.
D	Shift switch.
E	Alternate switch.
F	Small switch.
G	Insert switch.
K	Key data.
L	Low mark.
M	High mark.
N	NUM.
O	LEN.
P	Screen partition.
Q	Screen cursor position.
V	Line length.
M$	Data array.

INSIDE THE PC-1500

Norlin Rober, 407 North 1st Ave., Marshalltown, IA 50158 is back with lots of "inside" information for users of the Sharp PC-1500. (Virtually all of this information also applies to the Radio Shack PC-2.)

In this issue Norlin provides a recap of information relating to the use of RAM. In coming issues he will provide detailed information on the ROM. If you are a long time subscriber to *PCN* you are well acquainted with the type of valuable information Norlin is able to glean. If you are a new subscriber, we are sure you will marvel at the wealth of information served up by our long time PC *Super Sleuth!*

PC-1500 SYSTEM RAM

Addresses 7600-7C00 contain the Display Buffer, Input Buffer, fixed variables, and various other RAM required by the system.

Because of incomplete decoding, the memory located 7600-77FF may also be addressed as 7000-71FF, 7200-73FF, or 7400-75FF.

In the PC-1500, the memory located 7800-7BFF is also addressable as 7C00-7FFF. (In some versions of the PC-1500, this will be recall-only.)

In the PC-1500A, the area 7C00-7FFFF contains RAM, intended for storage of machine-language programs.

DISPLAY BUFFER

Each dot of the LCD is turned on by the presence of a '1' bit in a corresponding location in the Display Buffer. The upper four dots (U) in a column of the LCD are determined by one nybble, and the lower three dots (L) by another, as tabulated below.

Memory Address:	Columns controlled Lo nybble:	Hi nybble:	Memory Address:	Columns controlled Lo nybble:	Hi nybble:
7600	0 U	78 U	7700	39 U	117 U
7601	0 L	78 L	7701	39 L	117 L
7602	1 U	79 U	7702	40 U	118 U
7603	1 L	79 L	7703	40 L	118 L
7604	2 U	80 U	7704	41 U	119 U
7605	2 L	80 L	7705	41 L	119 L
. . .			. . .		
764C	38 U	116 U	774C	77 U	155 U
764D	38 L	116 L	774D	77 L	155 L

Display annunciators are set by bits as follows:

764E Misc: 01, BUSY; 02, SHIFT; 04, Katakana characters; 08, SMALL;
10, III; 20, II; 40, I; 80, DEF
764F Modes: 01, DE; 02, G; 04, RAD; 10, RESERVE; 20, PRO; 40, RUN

774E Unused (By the PC-1500 or CE-150).
774F Unused

FIXED STRING VARIABLES

7650-5F E$	76B0-BF K$	7750-5F P$	77B0-BF V$
7660-6F F$	76C0-CF L$	7760-6F Q$	77C0-CF W$
7670-7F G$	76D0-DF M$	7770-7F R$	77D0-DF X$
7680-8F H$	76E0-EF N$	7780-8F S$	77E0-EF Y$
7690-9F I$	76F0-FF O$	7790-9F T$	77F0-FF Z$
76A0-AF J$		77A0-AF U$	

SYSTEM RAM

7800-784F comprise the stack used by the CPU. Storage into this stack begins at 784F, continuing in reverse direction through memory.

7850 Unused
7851 Unused
7852 Unused
7853 Unused
7854 Unused
7855 Unused
7856 Unused
7857 Unused
7858 Unused
7859 Unused
785A Unused
785B High byte, address of external character input routine
785C Low byte, external character input routine
785D If 00, Katakana displayed; 80, displayed and printed; FF, neither
785E High byte, location of table of Katakana character codes
785F Unused
7860 High byte, START OF ROM in Module; FF if no Module Present.
7861 High byte, START OF BASIC Program in ROM Module; FF if no Module.
7862 Low byte, START OF BASIC Program in Module; FF if no Module.
7863 High byte, START OF RAM
7864 High byte, TOP OF RAM
7865 High byte, START OF BASIC Program in RAM
7866 Low byte, START OF BASIC Program in RAM
7867 High byte, END OF BASIC Program (stop byte address)
7868 Low byte, END OF BASIC Program
7869 High byte, START OF EDIT
786A Low byte, START OF EDIT
786B Flags: 01, BEEP OFF; 80, RMT ON
786C Unused
786D Unused
786E Unused
786F Unused
7870 Unused
7871 WAIT setting: 01, WAIT; 02, WAIT nnnn; 03, WAIT 0
7872 High byte, WAIT time counter
7873 Low byte, WAIT time counter
7874 Flags: 01, Cursor enabled; 80, display currently saved 7B10-7BAB
7875 CURSOR POINTER (current display column number)
7876 Character Position number in display, with INPUT statement
7877 Complement of number of display positions left for INPUT Prompt
7878 Specification for BEEP frequency
7879 Cassette Parameter: 1, MERGE; 2, CHAIN; 4, Data file; 8, M.L.; 10, RMT 1; 40, CLOAD?; 80, load operation
787A Unused

787B Position of blink character in display, plus 8
787C Flags: 01, blink cursor enabled: 80, a character is now blinked
787D Code of character blinked
787E High byte, location (in Display Buffer) of Blink Cursor
787F Low byte, location of Blink Cursor
7880 Display Parameter; determines display at READY, depending on bits set. 20, display contents of R0; otherwise, contents of Input Buffer. 10, display as BASIC line; 04, include colon after line number. 40, cursor activated in display. 80, ERROR message displayed. 08, RESERVE keyphrase displayed. 01, RESERVE template displayed, with previous display saved in 7B10-7BAB.
7881 Parsing Parameter; specifies type of instruction that will conform to rules of syntax for evaluating BASIC expression.
7882 Subpointer, used with GOSUB and FOR stacks; also, 01 indicates a USING statements that is not part of a PRINT or PAUSE statement.
7883 High byte, LET Pointer; also, 80 indicates no data pass with CALL.
7884 Low byte, LET Pointer; also used to store code of RESERVE key.
7885 With LET Pointer: length if string, 88 if numeric variable
7886 High byte, INPUT Pointer (starting address of variable)
7887 Low byte,INPUT Pointer
7888 With INPUT Pointer: length if string, 88 if numeric variable
7889 Flag: 01, variable subscript(s) being Parsed
788A Program Halt Parameter: 10, waiting for input; 20, halt for PRINT; 40, INPUT statement pending; 80, Break.
788B Low byte, Input Buffer Pointer
788C Number of function input arguments (used with Parsing routine).
788D TRACE: 00, OFF; other, ON. (Contents of 79D1 stored here by TRON)
788E TRACE Parameter: 00, starting execution of new line; 01, starting execution of a Program; 02, expression in Input Buffer is to be evaluated; 04, expression has been evaluated (immediate mode)
788F Low byte, Output Buffer Pointer
7890 Low byte, FOR stack Pointer
7891 Low byte, GOSUB stack Pointer
7892 Low byte, BASIC DATA STACK Pointer, used by Parsing routine
7893 Low byte, BASIC PENDING OP STACK Pointer, used by Parsing routine
7894 Low byte, String Buffer Pointer
7895 USING editing character: 10, comma separation; 20, forced sign; 40, asterisk fill; 80, scientific. (01 used to check syntax in interpretation of USING statement)
7896 USING number of characters,including sign, before decimal Point
7897 USING number of characters in string; 00, unspecified
7898 USING number of characters including and following decimal Point
7899 High byte, START OF VARIABLES in Main memory
789A Low byte, START OF VARIABLES in Main memory
789B Error Code
789C High byte, CURRENT line number; 00 if no Program in Progress
789D Low byte, CURRENT line number; 00 if no Program in Progress
789E High byte, beginning address of CURRENT Program
789F Low byte, beginning address of CURRENT Program
78A0 High byte, PREVIOUS address
78A1 Low byte, PREVIOUS address
78A2 High byte, PREVIOUS line number
78A3 Low byte, PREVIOUS line number
78A4 High byte, beginning of Program containing PREVIOUS line
78A5 Low byte, beginning of Program containing PREVIOUS line
78A6 High byte, SEARCH address
78A7 Low byte, SEARCH address

78A8 High byte, SEARCH line number
78A9 Low byte, SEARCH line number
78AA High byte, beginning of Program containing SEARCH line
78AB Low byte, beginning of Program containing SEARCH line
78AC High byte, BREAK address
78AD Low byte, BREAK address
78AE High byte, BREAK line number
78AF Low byte, BREAK line number
78B0 High byte, beginning of Program containing BREAK line
78B1 Low byte, beginning of Program containing BREAK line
78B2 High byte, ERROR address
78B3 Low byte, ERROR address
78B4 High byte, ERROR line number
78B5 Low byte, ERROR line number
78B6 High byte, beginning of Program containing ERROR line
78B7 Low byte, beginning of Program containing ERROR line
78B8 High byte, ON ERROR address
78B9 Low byte, ON ERROR address
78BA High byte, ON ERROR line number
78BB Low byte, ON ERROR line number
78BC High byte, beginning of Program containing ON ERROR line
78BD Low byte, beginning of Program containing ON ERROR line
78BE High byte, DATA Pointer; 80 indicates no DATA line yet located.
78BF Low byte, DATA Pointer

FIXED VARIABLES

78C0-CF A$	7920-27 E	7858-5F L	7998-9F T
78D0-DF B$	7928-2F F	7960-67 M	79A0-A7 U
78E0-EF C$	7930-37 G	7968-6F N	79A8-AF V
78F0-FF D$	7938-3F H	7970-77 O	79B0-B7 W
7900-07 A	7940-47 I	7978-7F P	79B8-BF X
7908-0F B	7948-4F J	7980-87 Q	79C0-C7 Y
7910-17 C	7950-57 K	7988-8F R	79C8-CF Z
7918-1F D		7990-97 S	

SYSTEM RAM <FOR OPTIONS>

79D0 ROM Bank: 00, ROM 1; 01, ROM 2
79D1 OPN device code: 60, LCD; 5C, CMT; 58, MGP; C4, LPRT; C0, COM
79D2 Unused
79D3 If 55, bypass setting of Modulation clock frequency, serial output
79D4 If 55, bypass keyboard scan; obtain input from external device
79D5 Unused
79D6 Unused
79D7 Unused
79D8 Unused
79D9 Used (by option)
79DA If 55, interrupt routine address in option is obtained from 79DB-D
79DB High byte, interrupt routine address in option
79DC Low byte, interrupt routine address in option
79DD Unused
79DE Unused
79DF Unused

79E0 High byte of X-coordinate, in signed binary (GRAPH mode); also used for high byte of address of first line to be LLISTed
79E1 Low byte of X-coordinate, in signed binary (GRAPH mode); also used for low byte of address of first line to be LLISTed
79E2 High byte of Y-coordinate, in signed binary (GRAPH mode); also, high byte of address of last line to be LLISTed
79E3 Low byte of Y-coordinate, in signed binary (GRAPH mode); also, low byte of address of last line to be LLISTed
79E4 High byte, Paper reverse feed count (counts from 0001 to 01FF)
79E5 Low byte, Paper reverse feed count
79E6 Location of Pen (00 to D8)
79E7 Low byte, extension of X-coordinate beyond available space
79E8 High byte, extension of X-coordinate beyond available space
79E9 Pen Parameter; specifies whether Pen is to be raised or lowered
79EA LINE TYPE (0 to 9), GRAPH mode
79EB Dotted-line counter
79EC Current Pen Position: 00, up; 01, down
79ED X-motor hold counter
79EE Motor Phase; stored into Port C
79EF Y-motor hold counter
79F0 Print Mode: 00, TEXT; FF, GRAPH
79F1 Printer disable: 0F, Pen change mode; FF, low btry or head blocked
79F2 ROTATE setting (0 to 3)
79F3 COLOR setting (0 to 3)
79F4 CSIZE setting (1 to 9)
79F5 LPRINT Parameter: 00, single-item; 04, comma LPRINT; 08, semi-colon LPRINT; 01, last item reached. Also used with LLIST for maximum Permissable line length.
79F6 With LINE, used as direction Parameter; with LLIST, to determine line feed; with COLOR, to save Pen location.
79F7 Type of data LPRINTed: 00, numeric; FF, character string
79F8 Temporary storage of Pen location during Paper feed
79F9 Flag: flag indicating PowerUP or interrupt in Progress
79FA Unused
79FB Unused
79FC Unused
79FD Unused
79FE Unused
79FF LOCK Mode: 00, LOCK; FF, UNLOCK

BASIC REGISTERS

Seven registers, used in BASIC computation, are located as follows:

7A00-07 R0 (Floating-Point accumulator)
7A08-0F R1 (Scratch register)
7A10-17 R2 (Second operand)
7A18-1F R3 (Scratch register)
7A20-27 R4 (Scratch register)
7A28-2F R5 (Scratch register)
7A30-37 R6 (Temporary storage register)

Each of these registers may contain a two-byte signed-binary number; a floating-point decimal number; or a pointer to a character string. The eight bytes in a register are utilized as follows:

BYTE	SIGNED BINARY	FLOATING-POINT	STRING POINTER
0	Unused	Exponent (binary)	Unused
1	Unused	Sign: 80 specifies -	Unused
2	Unused	1st & 2nd digits	Unused
3	Unused	3rd & 4th digits	Unused
4	B2 (to identify)	5th & 6th digits	D0 (to identify)
5	1st byte	7th & 8th digits	High byte, address
6	2nd byte	9th & 10th digits	Low byte, address
7	Unused	11th & 12th digits	Number of characters

Mantissas of calculated floating-point numbers are rounded to ten digits prior to display, printing, or storage. A string pointer formed by execution of CHR$ uses C1 instead of D0 as byte 4.

The area 7A00-7A37 is also used for other purposes, as follows:

The powerup routine uses addresses 7A10-14 (7A30-34 in earlier ROM versions) to temporarily save the addresses to be stored into the memory allocation pointers (RAM addresses 7860-64). Other pertinent information is stored by the powerup routine as follows:

7A20 If 00, powerdown was by 'OFF' key; 01, timed powerdown; 02, other type powerdown; 04, memory allocation pointers have been changed.
7A21 Set to other than zero by option with low battery power, indicating need for display of 'CHECK' message.
7A22 Set by option to indicate need for 'CHECK' message

A powerdown resulting from 7 minutes of non-use is identified by storage of A0, A1, ..., AF into addresses 7A10-7AFF; the stack pointer is saved in 7A30-31. (The powerdown initiated by use of the OFF key stores 50, 51, ..., 5F into 7A10-7AFF.)

The following are used in calculation of transcendental functions:

7A18 Flag: 00, SIN, COS, or ASN; 01, LOG, LN, or EXP; 20, ACS; 40, TAN or ATN; 80, result must be subtracted from 90 (inverse trig fctns)
7A20 Flag: 00, SIN; 01, COS

The PRINT and PAUSE routines use the area 7A10-7A34 to form a string of characters that represent numeric data formatted for display. The same area is used to form an ERROR or BREAK message.

The area beginning 7A08 is used as a scratch area in the interpretation of USING statements, and in execution of certain printer routines. In some of the printing routines, the contents of CPU register Y are saved in 7A26-7A27.

POCKET COMPUTER NEWSLETTER
P.O. Box 232, Seymour, CT 06483